Journal of Information Hiding and Multimedia Signal Processing ©2017 ISSN 2073-4212
Ubiquitous International Volume **8**, Number **3**, May **2017**

Path ω-automaton

Hua Jiang[1,2], Fumin Zou[2,4], Rongde Lin[3], and Lingxiang Li[5,*]

[1]Key Lab of Granular Computing, Minnan Normal University
Zhangzhou, Fujian, China

[2]Beidou Navigation and Smart Traffic Innovation Center of Fujian Province
Fuzhou, Fujian, China

[3]School of Mathematical Science, Huaqiao university
Quanzhou,Fujian, China

[4]Fujian Provincial Key Laboratory of Big Data Mining and Applications
Fuzhou, Fujian, China

[5]School of Electronics and Information Engineering
Hunan University of Science and Engineering, Yongzhou, Hunan, China

[*]Corresponding author: lilingxiang2013@hotmail.com

Received August, 2016; revised March, 2017

Abstract. *Automata theory is an important branch of theoretical computer science. ω-automaton is an important part of automata theory. The judgment standard for the equivalence of two automata is the equivalence of their acceptance language. Acceptance conditions for automata are expressed with state set. This paper tries to use path to express acceptance conditions for ω-automaton that is convenient for judging the automatons equivalence property and transforming between different types of automata. With the help of Path ω-automaton, Muller automaton is transformed into Büchi automaton. Compared with the existing algorithm, the state scale of Büchi automaton has been reduced and the problem of state space explosion has been relieved. The research findings in this paper contribute to enrich the existing ω-automaton theory and provide a new idea for ω-automaton theory and application research.*

Keywords: ω-automaton, Büchi automata, Muller automaton, transformation algorithms.

1. **Introduction.** Automata theory [1] is a branch of theoretical computer science. It studies self-operating virtual machines to help in logical understanding of input and output process, without or with intermediate stage(s) of computation (or any function / process). Finite automata on infinite objects were first introduced in the 60s [2–6]. Muller made use of automata on infinite words, named ω-automata [4], to describe problems in asynchronous switching theory. An ω-automaton (or stream automaton) is a variation of finite automaton which runs on infinite, rather than finite, strings as input. Because ω-automata do not stop, they have many kinds of acceptance conditions rather than simply a set of accepting states. ω-automata are helpful for specific systems behavior which are not expected to end, such as hardware, operating systems and control systems. Categories of ω-automata contain Büchi automata [2], Rabin automata [6], Streett automata [7], parity automata [8] and Muller automata [4], each deterministic or non-deterministic. These categories of ω-automata are different only in terms of acceptance conditions. Except for deterministic Büchi automata, ω-automata can be transformed into other types of

ω-automata [5, 9–13]that have the same acceptance language. In other words, except for deterministic Büchi automata, ω-automata have the same describing ability.

A Muller automaton $A = (Q, \Sigma, \delta, q_0, AccS)$, Let $|Q| = n, |AccS| = m, |\delta| = O(n^2)$. In [13] the authors construct a Büchi automaton out of an EL automaton resulting in an automaton of size $O(mn^2)$, but the EL automaton is a slightly different way to specify a Muller automaton. In [14], there is an algorithm for a nondeterministic Muller automaton which can be transformed into an equivalent Büchi automaton with $O(mn2^n)$ states. [15] presented a construction with a detour via regular expressions,the Büchi automaton's size is bounded by $|Q| * |\delta| * |AccS| = O(mn^3)$.

This paper presents Path ω-automaton, a new expressing method of ω-automaton. Using this describing method, other types of ω-automata can be transformed into Path ω-automaton. In the end,this paper studies an algorithm for transforming nondeterministic Muller automaton into nondeterministic Büchi automaton. Through our research we discover that the algorithm in this paper can directly transform a nondeterministic Muller automaton with n states and m accepting sets into an equivalent Büchi automaton with $O(mn^2)$ states. Compared with the existing algorithm of the same kind, the algorithm in this paper is a direct transformation algorithm, which has an easy transformation operation. For the upper bound of the state numbers of transformed Büchi automaton, this algorithm in this paper is better than others.

ω-automata has been the research hotspot in computer theory all around the world. Part of recent researches in this field can be seen in reference [12, 16–21] and [22]. Path ω-automaton proposed in this paper is expected to further enrich the theory of ω-automaton and provides a new perspective and method for theory research and practical application of ω-automaton.

2. **Path ω-automaton.**

2.1. **Basic concept.**

Definition 2.1. *[23] An ω-automaton is a quintuple$(Q, \Sigma, \delta, q_0, AccS)$, where Q is a finite set of states, Σ is a finite set of alphabets, $\delta : Q \times \Sigma \to 2^Q$ is the state transition function ,$q_0 \in Q$ is the inital state, and $AccS$ is the acceptance component.In a deterministic ω-automaton, a transition function $\delta : Q \times \Sigma \to 2^Q$ is used.*

Definition 2.2. *[23] For an ω-automaton $A = (Q, \Sigma, \delta, q_0, AccS)$, a run of A on an ω-word $\alpha = \alpha(1)\alpha(2)... \in \Sigma^\omega$ is a finite state sequence, $\pi = \pi(1)\pi(2)... \in Q^\omega$, then the following conditions hold:*

(1)$\pi(0) = q_0$,
(2)$\pi(i) \in \delta(\pi(i-1), \alpha(i)) forall\ i \geq 1$ if A is nondeterministic,
$\pi(i) = \delta(\pi(i-1), \alpha(i)) forall\ i \geq 1$ if A is deterministic

Let $Inf(\alpha) = \{a \in \Sigma | \forall i, \exists j > i.\alpha(j) = a\}$

Definition 2.3. *[2] An ω-automaton $A = (Q, \Sigma, \delta, q_0, AccS)$ with acceptance component $AccS \in Q$ is called Büchi automaton if it is used with the following Büchi acceptance condition: A word $\alpha \in \Sigma^\omega$ is accepted by A iff there exists a run π of A on α satisfying the condition $Inf(\pi) \cap AccS \neq \emptyset$, i.e. at least one of the states in $AccS$ has to be visited infinitely often during the run. Let $L(A) := \{\alpha \in \Sigma^\omega | A$ accepts $\alpha\}$ is the ω-language recognized by A.*

Definition 2.4. *[4] An ω-automaton $A = (Q, \Sigma, \delta, q_0, AccS)$ with acceptance component $AccS \in 2^Q$ is called Muller automaton if it is used with the following Muller acceptance condition: A word $\alpha \in \Sigma^\omega$ is accepted by A iff there exists a run π of A on α satisfying* *the condition $Inf(\pi) \in AccS$, i.e. the set of infinitely recurring states of π is exactly one of the sets in $AccS$.*

Obviously, if a word $\alpha \in \Sigma^\omega$ is accepted by A, there exists a run π of A on α satisfying the condition $Inf(\pi) \in AccS$, then the subgraph constructed by $Inf(\pi)$ and the transitions over it is a strongly connected subgraph.

2.2. **Path ω-automaton.** If ω-automaton A and B are equivalent, it means that the acceptance language of A and B are accordant. It defines the equivalence from the view of automaton running and it is a dynamic definition, while the acceptance condition for commonly defined automaton is expressed by state set or state superset and it is a static definition. Although statically defined ω-automaton is more close to the need of practical application, dynamic definition is easier to be dealt with from the views of language recognized by automaton, transformation between automata and the equivalence of automaton, which can be clearly known in section 3.

Let α, β... are regular expressions, $\alpha \cdot \beta$ means that the destination of α is the same with the starting point of β, $\alpha \cdot \beta$ is a path connected by α and β.

A dynamic definition of ω-automaton, Path ω-automaton, is given as follows:

Definition 2.5. *A Path ω-automaton $A = (Q, \Sigma, \delta, q_0, \Omega)$ where Q is a finite set of states, Σ is a finite set of alphabets, $\delta : Q \times \Sigma \to 2^Q$ is the state transition function ,$q_0 \in Q$ is the inital state, and $\Omega = \{(E_1, F_1), ..., (E_r, F_r)\}$ is the acceptance component, E_i, F_i are sets of regular expressions, for $\forall 1 \le i \le r, F_i \ne \emptyset$. For $\forall 1 \le i \le r$, if $\alpha.\beta \wedge \alpha \in E_i \wedge \beta \in F_i$, then ω-word $\alpha \cdot \beta^\omega$ is accepted by A, and if an ω-word $\alpha \cdot \beta^\omega$ is accepted by A, then $\exists\ 1 \le i \le r, \alpha \cdot \beta \wedge \alpha \in E_i \wedge \beta \in F_i$*

Obviously, automaton A is defined with Path ω-automaton, all ω-word $\alpha \cdot \beta^\omega$ that satisfy Definition 2.5 constitute acceptance language $L(A)$ of automaton A.

In ω-automaton, there possibly exists many paths between two states, and an ω-word $\alpha.\beta^\omega$ also can be written as $\alpha{\cdot}\beta^*{\cdot}\beta^\omega$. Using Definition 2.5 to describe acceptance conditions directly, but E and F in (E, F) will be very complicated. To be more simplified, appoint $\forall 1 \le i \le r$,$E_i \ne \emptyset$, define $p(Q_1, Q_2, Q_3)$ to express the path whose starting point is $qs \in Q_1$ and terminal point is $qe \in Q_3$ in ω-automaton, and satisfy either reaching directly or only passing the states in Q_2.

Example 2.1. *The path that state q_1 directly reaches q_2 can be expressed with $p(\{q_1\}, \emptyset, \{q_2\})$, all the paths from state q_1 to q_2 can be expressed with $p(\{q_1\}, Q, \{q_2\})$.*

Example 2.2. *All the paths that start from q_1 , through q_2 , and finally reach q_3, can be denoted as $p(\{q_1\}, Q, \{q_2\}) \cdot p(\{q_2\}, Q, \{q_3\})$.*

Example 2.3. *Use Path ω-automaton to express a Büchi ω-automaton. A Büchi ω-automaton $A = (Q, \Sigma, \delta, q_0, AccS)$, $AccS = \{q_1, ..., q_r\}$, the equivalent Path ω-automaton is $A' = (Q, \Sigma, \delta, q_0, \Omega)$,$\Omega = \{(E_i, F_i)|1 \le i \le r \wedge E_i = \{p(\{q_0\}, Q, \{q_i\})\} \wedge F_i = \{p(\{q_i\}, Q, \{q_i\})\} \wedge q_i \in AccS\}$.*

Example 2.4. *Use Path ω-automaton to express a Rabin ω-automaton. A Rabin ω-automaton $A = (Q, \Sigma, \delta, q_0, \Omega)$, $\Omega = \{(L_1, U_1), ..., (L_r, U_r)\}$, the equivalent Path ω-automaton is $A' = (Q, \Sigma, \delta, q_0, \Omega')$,$\Omega' = \{(E', F')|\forall 1 \le i \le r, E' = \{p(\{q_0\}, Q, \{U_i \backslash L_i\})\} \wedge F' = \{p(\{U_i \backslash L_i\}, Q \backslash L_i, \{U_i \backslash L_i\})\} \wedge E'_i \cdot F'_i \wedge (L_i, U_i) \in \Omega\}$.*

Example 2.5. *Use Path ω-automaton to express a Streett ω-automaton. A Streett ω-automaton $A = (Q, \Sigma, \delta, q_0, \Omega)$, $\Omega = \{(E_1, F_1), ..., (E_r, F_r)\}$, the equivalent Path ω-automaton is $A' = (Q, \Sigma, \delta, q_0, \Omega')$,$\Omega' = \{(E', F')|\forall 1 \le i \le r, E' = \{p(\{q_0\}, Q, \{E_i\})\} \wedge F' =$*

$\{p(\{E_i\},Q,\{E_i\})\}$ or $E' = \{p(\{q_0\},Q,\{Q\backslash F_i\})\} \wedge F' = \{p(\{Q\backslash F_i\},Q\backslash F_i,\{Q\backslash F_i\})\} \wedge E_i \cdot F_i \wedge (E_i,F_i) \in \Omega\}$.

Example 2.6. *Use Path ω-automaton to express a Parity ω-automaton. A Parity ω-automaton $A = (Q,\Sigma,\delta,q_0,c), c : Q \to \{0,1,...,k\}$,the equivalent Path ω-automaton is $A' = (Q,\Sigma,\delta,q_0,\Omega)$,$\Omega = \{(E,F)|1 \le i \le \lfloor k/2 \rfloor \wedge E = \{p(\{q_0\},Q,\{F_i\backslash E_i\})\} \wedge F = \{p(\{F_i\backslash E_i\},Q\backslash E_i,\{F_i\backslash E_i\})\} \wedge E_i \cdot F_i \wedge \{(E_i,F_i)|E_i = \{q \in Q|c(q) \le 2i\}, F_i = \{q \in Q|c(q) \le 2i+1\}\}\}$.*

Lemma 2.1. *An Muller ω-automaton $A_1 = (Q,\Sigma,\delta,q_0,AccS), AccS = \{Q_1,...,Q_r\}, Q_i = \{q_{i1},...,q_{it}\}$,A Path ω-automaton is $A_2 = (Q,\Sigma,\delta,q_0,\Omega)$,$\Omega = \{(E_i,F_i)|1 \le i \le r \wedge E_i = \{p(\{q_0\},Q,\{q_{i1}\})\} \wedge F_i = \{p(\{q_{i1}\},Q_i,\{q_{i2}\}) \cdot p(\{q_{i2}\},Q_i,\{q_{i3}\}) \cdot ... \cdot p(\{q_{it}\},Q_i,\{q_{i1}\})\}$; then $L(A_1) = L(A_2)$.*

Obviously, all paths of F_i in A_2 pass all the states in Q_i and pass the states in Q_i only; it is in accordance with the definition of acceptance conditions for Muller ω-automaton in Definition 2.4, it is equivalent to A_1 in acceptance language. Lemma 2.1 shows that Muller ω-automaton and Path ω-automaton have the same acceptance language and describing ability.

From the above examples we know that no matter how the acceptance conditions define, it can be easily transformed into Path ω-automaton, which provides the transformation among different types of ω-automaton with a fast way.

3. Muller automaton transforming into Büchi automaton.

According to Lemma 2.1 and Example 2.3, both Muller automaton and Büchi automaton can be expressed with Path ω-automaton. The transformation algorithm from a Muller automaton $A = (Q,\Sigma,\delta,q_0,AccS)$ into the equivalent Büchi automaton $A' = (Q',\Sigma,\delta',q_0,AccS')$ is presented in Algorithm 3.1.

Algorithm 3.1. Let $m = |AccS|, AccS = \{S_1,S_2,...,S_m\}, S_h = \{q_{(h,1)},q_{(h,2)},...,q_{(h,x)}\}, x = |S_h|(1 \le h \le m)$, and

```
1. $Q' = Q, \delta' = \delta, h = 1$
2. Copy the states in $S_h = \{q_{(h,1)},q_{(h,2)},...,q_{(h,x)}\}$, rename it as $q'_{(h,1)},q'_{(h,2)},...,q'_{(h,x)}$ , and add the states after renaming to $Q'$
3. Copy the states in $S_h = \{q_{(h,1)},q_{(h,2)},...,q_{(h,x)}\}$ for $x$ backups, rename them as the following rules, the backups of $\forall q_{(h,j)} \in S_h, q_{(h,j)}$ can be renamed as $q_{(h,j,1)},q_{(h,j,2)},...,q_{(h,j,x)}$ respectively, and add the states after renaming to $Q'$
4. $\forall a \in \Sigma$
   $for(i = 1; i \le x; i++)$
     $for(j = 1; j \le x; j++)$
       $if(\delta(q_{(h,i)},a) = q_{(h,j)})$
         { $\delta' = \delta' \cup \{\delta'(q_{(h,i,k)},a) = q'_{(h,j,k)}|1 \le k \le x\} \cup \{\delta'(q'_{(h,i)},a) = q_{(h,j,i)}\}$
          $if(j = 1)\ \delta' = \delta' \cup \{\delta'(q_{(h,i)},a) = q'_{(h,j)}\} \cup \{\delta'(q_{(h,i,x)},a) = q'_{(h,j)}\}$
          $else\ \delta' = \delta' \cup \{\delta'(q_{(h,i,j-1)},a) = q'_{(h,j)}\}$
         }
5. for $2 \le h \le m$, repeat step 2 to step 4.
6. $AccS' = \{q'_{(i,1)}|1 \le i \le m\}$.
```

By Algorithm 3.1, a nondeterministic Muller automaton $A = (Q,\Sigma,\delta,q_0,AccS)$ can be transformed into a nondeterministic Büchi automaton $A' = (Q',\Sigma,\delta',q_0,AccS')$. An example of a Muller automaton transforming into Büchi automaton by Algorithm 3.1 is shown in Example 3.1.

Example 3.1. *A Muller automaton transforms into Büchi automaton by Algorithm 1 is shown in Fig.1.*

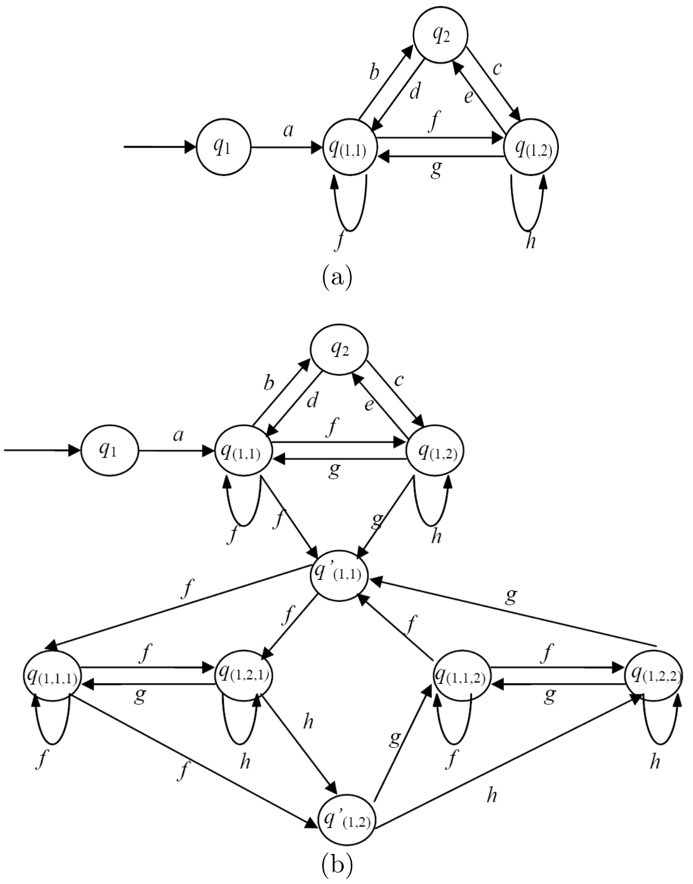

FIGURE 1. Muller automaton transforms into Büchi automaton by Algorithm 1. (a) Muller automaton ($AccS = \{\{q_{(1,1)}, q_{(1,2)}\}\}$). (b) Büchi automaton($AccS' = \{q'_{(1,1)}\}$).

In Example 3.1, the Muller automaton accepts $a.(f.g)^\omega$, Büchi automaton in this example exists acceptable path $q_1 q_{(1,1)} q_{(1,2)} (q'_{(1,1)} q_{(1,2,1)} q_{(1,1,1)} q'_{(1,2)} q_{(1,1,2)} q_{(1,2,2)})^\omega$. The word corresponding to the path is $a \cdot f \cdot g \cdot (f \cdot g \cdot f \cdot g \cdot f \cdot g)^\omega$, and it is obviously equivalent to $a \cdot (f \cdot g)^\omega$.

Theorem 3.1. *By Algorithm 3.1, a nondeterministic Muller automaton $A = (Q, \Sigma, \delta, q_0, AccS)$ can be transformed into the nondeterministic Büchi automaton $A' = (Q', \Sigma, \delta', q_0, AccS')$, then $L(A) = L(A')$.*

Proof. For a nondeterministic Muller automaton $A = (Q, \Sigma, \delta, q_0, AccS)$, based on Definition 2.4,a word $\alpha \in \Sigma^\omega$ is accepted by A iff there exists a run π of A on α satisfying the condition: $Inf(\pi) \in AccS$.

Any $\alpha \in L(A)$ and corresponding π satisfy $Inf(\pi) = S_i$, α can be divided into two parts: infinite and finite. Infinite part only passes the states in S_i and the states in S_i are passed for infinite times. Finite part is the rest of α that removes infinite part; it is the prefix of infinite part.

For finite part, because the transition diagram of Büchi automaton $A' = (Q', \Sigma, \delta', q_0, AccS')$ reserves all part of A, it can simulate finite part in A'.

For infinite part, take a finite fragment in α that satisfies the starting point is $q_{(i,j)}$, terminal point is $q_{(i,j+1)}$ and length > 1(if length=1, then take $q_{(i,j+1)}$ that appears next time.), it only passes the states in S_i. In A', the corresponding subgraph to $q_{(i,1,j)}, q_{(i,2,j)}, ..., q_{(i,x,j)}$ completely copies the corresponding subgraph to S_i, the fragment can be simulated by A'. Take any state $q_{(i,j)}$ in S_i, if visiting $q_{(i,j)}$ for infinite times, α must contain infinite incident arc and outgoing arc(all or part) of $q_{(i,j)}$. Observe $q'_{(i,j)}$ in A', word contains infinite incident arc and outgoing arc of $q'_{(i,j)}$ and visits $q'_{(i,1)}$ for infinite times, $q'_{(i,1)} \in AccS'$. According to the definition of Büchi automaton, $\alpha \in L(A'), L(A) \subseteq L(A')$.

Let $\alpha' \in L(A')$ and the corresponding π' satisfies $q'_{(i,1)} \in Inf(\pi')$, α' can be divided into two parts: infinite part and finite part. Finite part corresponds to the part before reaching state $q'_{(i,1)}$ that can be completely simulated by A. For infinite part, $q'_{(i,1)} \in Inf(\pi')$, because word α' needs to pass $q'_{(i,1)}$ for infinite times, it must pass $q'_{(i,2)}, ..., q'_{(i,x)}$ for infinite times, $q'_{(i,2)} \in Inf(\pi'), ..., q'_{(i,x)} \in Inf(\pi')$.Take the finite fragment in α' that satisfies the starting point is $q'_{(i,j)}$ and terminal point is $q'_{(i,j+1)}$, the corresponding subgraph to $q_{(i,1,j)}, q_{(i,2,j)}, ..., q_{(i,x,j)}$ completely copies the corresponding subgraph to S_i ,so the fragment can be simulated by the corresponding subgraph to S_i. Any fragment in the infinite part of α' can be simulated by S_i. Because α' needs to pass $q'_{(i,j)} 1 \leq j \leq |S_i|$ for infinite times, α' must contain infinite incident arc and outgoing arc of $q'_{(i,j)}$ (all or part); so α' can be simulated by A, it must visit $q_{(i,j)}$ for infinite times, i.e. α' can be simulated by A, it must visit all states in S_i for infinite times. According to the definition of Muller automaton, $\alpha' \in L(A), L(A') \subseteq L(A)$.

Therefore $L(A) = L(A')$ □

Theorem 3.2. *A nondeterministic Muller automaton $A = (Q, \Sigma, \delta, q_0, AccS)$, where Q has n states and $AccS$ contains m sets, can be transformed into an equivalent nondeterministic Büchi automaton $A' = (Q', \Sigma, \delta', q_0, AccS')$, with $O(mn^2)$.*

Proof. Let $|Q| = n, |AccS| = m$, by step 1 of Algorithm 3.1, $|Q'| = n$; and by step 2, $|Q'| \leq n + mn$; and furthermore by step 3, $|Q'| \leq n + mn + mn^2 = O(mn^2)$. □

Theorem 3.3. *A nondeterministic Muller automaton $A = (Q, \Sigma, \delta, q_0, AccS)$, where Q has n states and $AccS$ contains m sets, can be transformed into an equivalent nondeterministic Büchi automaton $A' = (Q', \Sigma, \delta', q_0, AccS')$, with $O(mn^2)$ by Algorithm 3.1, and the time complexity is $O(mn^3)$.*

Proof. In Algorithm 3.1, the core operation is copying subgraphs that correspond to all elements in $AccS$. There are n^2 transition relations at most in each subgraph, and each subgraph can be copied for n times. There are m subgraphs to be copied, so the time complexity needed for copying subgraph corresponds to all sets in $AccS$ is $O(m.n^3)$. □

The efficiency of Algorithm 3.1 can be further improved on some technical aspects: Muller automaton A can be simplified, some unnecessary operations can be reduced, and transformation computations can be more efficiently, which are shown as the following:

1. Any unreachable state from q_0 can be removed from Q, the corresponding transition relations can be removed from δ.

2. Any state that starts from this state and cannot reach accepting state can be removed from Q, the corresponding transition relations can be removed from δ.

3. For a certain acceptance set in $AccS$, if the state and the transition relations adhered to it cannot constitute a strongly connected subgraph, this acceptance set can be removed from $AccS$.

4. When a certain acceptance subset $|S_h| = 1$, skip step 3 in Algorithm 3.1, and step 4 is modified as follows:

```
$if(x = 1)$
  $\forall a \in \Sigma$
    $if(\delta(q_{(h,1)}, a) = q_{(h,1)})$
      $\delta' = \delta' \cup \{\delta'(q_{(h,1)}, a) = q'_{(h,1)}\} \cup \{\delta'(q'_{(h,1)}, a) = q_{(h,1)}\}$
```

Example 3.2. *Given Muller automaton in Fig 2, if $AccS = \{\{q_2\}, \{q_3, q_4\}\}$, because $\{q_3, q_4\}$ and the transition relations adhered to it cannot constitute a strongly connected subgraph, it can be removed from $AccS$, the acceptance component in Muller automaton can be simplified as $AccS = \{\{q_2\}\}$.*

Example 3.3. *Fig.3(a) shows a Muller automaton with acceptance subset $|S_h| = 1$ ($AccS = \{\{q_{(1,1)}\}\}$) , so the step 3 of Algorithm 3.1 can be skipped,and use the step 4 of improved algorithm makes the equivalent Büchi automaton shown in Fig.3(b).*

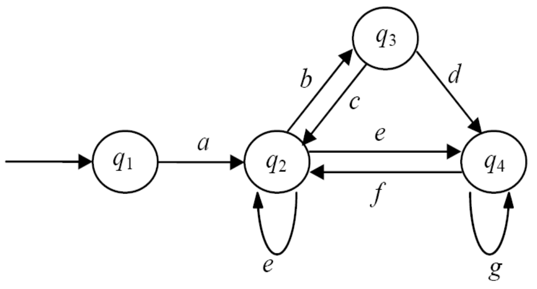

FIGURE 2. Muller automaton ($AccS = \{\{q_2\}, \{q_3, q_4\}\}$)

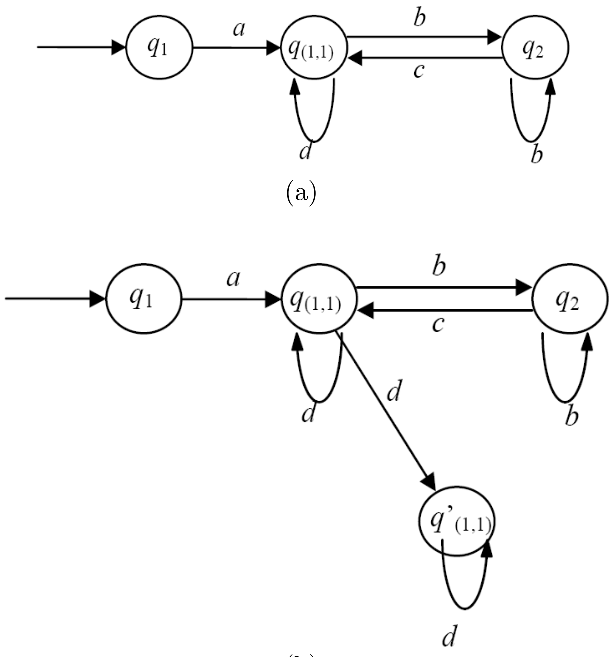

FIGURE 3. Muller automaton transforms into Büchi automaton by improved Algorithm 3.1. (a) Muller automaton ($AccS = \{\{q_{(1,1)}\}\}$). (b) Büchi automaton($AccS = \{q'_{(1,1)}\}$).

4. **Conclusion.**

Path ω-automaton put forward in this paper is an expressing way of path method for ω-automaton. Compared with the definition of existing ω-automaton, although acceptance conditions for the definition are a little complicated, using Path ω-automaton to express nondeterministic Muller automaton and nondeterministic Büchi automaton, the algorithm for transforming nondeterministic Muller automaton into nondeterministic Büchi automaton can be found easily.

In this paper, we use nondeterministic Muller automaton to construct a language equivalent nondeterministic Büchi automaton and get a new Büchi automaton whose state scale has polynomial relationship to the state scale and the accepted set scale of nondeterministic Muller automaton. Compared with the existing transformation algorithm of the same kind, Büchi automaton got from the algorithm in this paper has the least state scale upper bound. We believe that Path ω-automaton will enrich the existing theory of ω-automaton and provide a new perspective and method for studying ω-automaton.

Acknowledgment. This work is supported by National Nature Science Foundation of China (No.61472406), Nature Science Foundation of Fujian
(No.2015J01269 and No.2016J01304), and the Talent Introduction Foundation of Minnan Normal University.

REFERENCES

[1] J. E. Hopcroft, Introduction to automata theory, languages, and computation, *[M]. Education India*,1979.

[2] R. Bchi, On a decision method in restricted second order arithmetic, *International Congress on Logic, Method and Philos. Sci.* , pp. 1-11, 1962.

[3] B. A. Trakhtenbrot. Finite automata and the logic of one-place predicates, *Sibirian Mathmatical Journal (English translation in: AMS Transl.*, vol.3, pp.103-131, 1962.

[4] D. E. Muller, Infinite sequences and finite machines *Switching Circuit Theory and Logical Design,Proceedings of the Fourth Annual Symposium on. IEEE,* pp. 3-16, 1963.

[5] R. McNaughton, Testing and generating infinite sequences by a finite automaton, *Journal of Information and control*, vol. 9, no. 5, pp. 521-530, 1966.

[6] M. O. Rabin ,Decidability of second-order theories and automata on infinite trees, *Journal of Transactions of the American Mathematical Society,* , pp. 1-35, 1969.

[7] R. S. Streett, Propositional dynamic logic of looping and converse is elementarily decidable, *Journal of Information and control*, vol. 54, no. 1, pp. 121-141, 1982.

[8] A. W. Mostowski, Regular expressions for infinite trees and a standard form of automata, *Lecture Notes in Computer Science,* vol. 208, pp. 157-168, 1984.

[9] S. Miyano, T. Hayashi, Alternating finite automata on w-words, *Journal of Theoretical Computer Science,* vol. 32, no. 3, pp. 321-330, 1984.

[10] O. Kupferman, M. Vardi, Weak Alternating Automata Are Not That Weak, *Theory of Computing Systems, Israel Symposium on the. IEEE Computer Society,* pp. 147-147, 1997.

[11] S. Safra, On the complexity of w-automata, *Foundations of Computer Science, 1988., 29th Annual Symposium on. IEEE*, pp. 319-327, 1988.

[12] C. Tian, Z. Duan, Büchi Determinization Made Tighter, *Journal of arXiv preprint arXiv,* pp. 1404.1436, 2014.

[13] S. Safra and M.Y. Vardi, On w-automata and temporal logic, *21st ACM Symposium on Theory of Computing,* pp. 127137, Seattle, May 1989.

[14] E. Grädel, Automata, Logics, and Infinite Games, *LNCS 2500, Springer 2002*, pp. 3-21.

[15] D. Perrin, J. Pin, Infinite words: Automata, Semigroups, Logic and Games, *Pure and Applied Mathematics*, vol. 141, Elsevier, 2004.

[16] S. Schewe, T. Varghese, Determinising Parity Automata, *Journal of arXiv preprint arXiv:1401.5394*, 2014.

[17] M. Skrzypczak, Descriptive set theoretic methods in automata theory, *Journal of [Online].* Available: `http://www.mimuw.edu.pl/~mskrzypczak/docs/sk14_thesis.pdf`,2014.

[18] Z. W. Han, Y. M. Li, Quantum Mller automata and monadic second-order quantum logic, *Journal of Ruan Jian Xue Bao/Journal of Software*, vol. 25, no. 1, pp. 2736 (in Chinese), 2014.

[19] J. Klein, C. Baier, S. Klppelholz, Compositional construction of most general controllers, *Acta Informatica.* DOI 10.1007/s00236-015-0239-9. Springer-Verlag Berlin Heidelberg 2015.

[20] F. Song, T. Touili, Model checking dynamic pushdown networks, *Formal Aspects of Computing,* vol. 27, no. 2, pp. 397-421, 2015.

[21] R. Chadha, A. P. Sistla, M. Viswanathan, Y. Ben, Decidable and Expressive Classes of Probabilistic Automata, *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, vol. 9034, pp. 200-214, 2015.

[22] E. Filiot, Logic-Automata Connections for Transformations, *Logic and Its Applications, Lecture Notes in Computer Science V8923*, pp. 30-57, 2015.

[23] C. Löding, Methods for the transformation of ω-automata: Complexity and connection to second order logic, *Diplomata thesis, Christian-Albrechts-University of Kiel*, 1998.